Contract Authority Under AI

I pursued the strongest claim I could make about contracts: that they could become the governing authority artifact for AI-assisted software work.

Not just helpful design documents. Not just better specs. Not just coordination tools. I wanted to know whether a sufficiently explicit contract could become the thing that implementation, testing, gating, and maybe even refusal ultimately answer to.

After months of pushing on that idea, I can't support that claim.

What I can support is narrower. Explicit source-of-truth artifacts helped a great deal. They improved AI-assisted iteration. They made important invariants easier to preserve. But they did not become the actual authority surface.

That distinction ended up mattering more than I expected.

How this started

I did not start with a grand theory about contracts. I got there because AI made it practical to push much more detail into written artifacts, and because I was already seeing what happened without explicit authority artifacts: code failed outright, reversions happened, and fixes were later wiped out when the AI misapplied my intent or the scope of the change. That pressure pushed looser documents into much more explicit contracts.

I was also working on real engineering systems with a real team, and we had a real coordination problem. Loose design notes were not enough. We needed one place that said how the system was supposed to work, what the important rules were, what the boundaries were, what shapes were allowed, and what could not be allowed to drift.

As the systems got more complicated and the coordination burden got heavier, those looser specs had to become something more precise. AI accelerated that shift for me. It made it practical to put more detail into the written artifact, tighten ambiguity faster, and keep refining the contract as both the system and the implementation work evolved.

I pushed this approach twice on two different serious systems. The second contract ran to 22 pages before counting the RFC or the ticketable work. These were not toy documents written to make a point. They were detailed because the systems were detailed. They had real business rules, real state guarantees, real replay concerns, real authority boundaries, and real consequences if those things were misunderstood.

And the results were very good.

The contracts helped people move faster. They helped implementations line up better. They helped engineers work against the same description of the system. They helped turn important business rules into something explicit instead of leaving them buried in memory, Slack, or tribal knowledge.

Then I started using those same contracts to drive AI-assisted implementation, and the results were strong enough, multiple times, that I thought there might be something much bigger here.

That is what led to the real question: not whether contracts were useful, but what exactly they were useful for.

What held up

The first thing that held up was simple.

AI-assisted work got better when there was a clear artifact outside the prompt and outside the code that said what the system was supposed to do and what had to stay true.

When I gave AI a detailed contract, the generated code matched the intended behavior more often. Important rules were less likely to get dropped. It was easier to derive tests and related artifacts from the same source. It was easier to iterate without losing the thread of the system.

That was not just true in one mode. I saw it in team coordination, solo rapid iteration, and multi-agent implementation. I saw it across repeated use, not just in one lucky run.

One answer that emerged pretty clearly was iterative stability.

The contracts helped most when a change had to hold across more than one surface. They made it easier to preserve important invariants when the change was not just one local edit but something that had to land in multiple places. They made cross-surface drift easier to see. That part is real, and I still think it matters.

So this is not a “contracts are useless” piece. It is not even close to that.

The narrower result survived.

The strongest claim

But that narrower result was not the strongest claim I was testing.

The strongest claim was that contracts could become the actual authority surface for AI-assisted software work. Not just support artifacts. Not just coordination artifacts. Not just inputs to better iteration. I wanted to know whether the contract itself could become the thing the whole system ultimately answers to.

That is the claim I tested.

And that claim did not survive.

Where it broke

The break did not happen because the contracts were vague. It did not happen because the work was shallow. And it did not happen because the contracts only helped in toy settings.

The break happened after the contracts became extremely explicit and after they had already proven themselves useful in serious engineering work.

That matters, because a document can help iteration without being authority. A document can improve alignment without being authority. A document can make important rules more visible without being authority. A document can even help block bad changes without being authority.

The question was whether the contract itself could become the place where the system really says yes, no, or stop.

I could not prove that.

And once I tried to force the contracts into that role, the mechanics started showing why.

The ai-contract-gating repo made that visible in a small, explicit form. It showed that an external normative artifact can feed a repeatable gating path. A pull request can be blocked because it violates preserved truths outside the implementation itself. That part is real.

But the same repo also showed the limit. Once the system actually worked, the contract was no longer the thing doing the live work. The live work had moved into the manifest, the fixture corpus, and the Python evaluator. The contract still mattered upstream, but the effective control in the system was downstream in the derived machinery.

That is the translation problem.

A contract does not become authority just because it can be translated into a downstream gate. The moment I needed a manifest, fixture set, evaluator, or other enforcement layer, the contract stopped being the place where the live decision was happening. It still mattered. It still carried the governing truth. But the actual yes, no, or stop had moved into the translated machinery.

Every step from contract to manifest, contract to evaluator, or contract to gate creates another place where intent can leak, be weakened, or be misread. And if AI is the thing doing that translation, the problem gets sharper, not softer. The same probabilistic system being guided by the contract is also being asked to build the layer that is supposed to constrain it.

That is why the downstream path never solved the original problem. The contract remained important, but it did not remain the authority surface. It became upstream input to a synthetic testing or enforcement layer.

I then tried the next obvious move. If the contract was not directly usable as the authority surface, maybe I could derive the enforcement layer from it. Maybe the contract could stay primary while the downstream pieces stayed thin and subordinate.

That also broke down.

Outside the narrow proof-of-concept, the pipeline degraded quickly. The outputs got weaker. They got less specific. They got less useful. And even when they were usable, they still depended on their own layer of machinery to matter.

At that point the conclusion stopped being theoretical. If a contract-derived artifact still needs its own synthetic implementation, harness, evaluator, fixtures, or runtime layer just to interact with the actual system, then it is not authority. It is another testing layer. Maybe a disciplined one. Maybe an interesting one. But still a testing layer.

And in the cases I pushed on, it was often less useful than simpler direct testing would have been.

What this gives away

I also need to be candid about what this gives away to the anti-spec side of the argument.

It absolutely gives away part of the strongest pro-contract pitch.

I cannot honestly say that contracts solve authority. I cannot honestly say that turning them into manifests and evaluators creates a fundamentally new control surface. I cannot honestly say that contracts, by themselves, are the answer to implementation, testing, gating, and refusal.

That criticism lands.

But that does not mean the anti-spec position wins outright either. The work does not support the idea that explicit source-of-truth artifacts are useless. It does not support the idea that AI iteration works just as well without them. If anything, it supports the opposite.

What failed was the strongest claim about what those artifacts could become.

That is the boundary of the work as I can honestly support it now. I can still defend contracts as valuable source-of-truth artifacts. What I cannot defend is the claim that I proved they become authority under collapsing implementation costs brought by AI-assisted coding.